


IMMEDIATE GENERATING METHOD

Patent Number: JP8194626
Publication date: 1996-07-30
Inventor(s): ISOBE TATSUO; KASHIWAGI YUUGO
Applicant(s):: HITACHI LTD; HITACHI SOFTWARE ENG CO LTD
Requested Patent:  JP8194626
Application Number: JP19950022311 19950117
Priority Number(s):
IPC Classification: G06F9/45
EC Classification:
Equivalents:

Abstract

PURPOSE: To enable the designation of an immediate exceeding a limit without adding any instruction for immediate generation.

CONSTITUTION: An immediate #Z not exceeding a first instruction with an immediate #X exceeding a designated frame as an operand is defined as a first operand and a register held value Y remaining in any register is defined as a second operand for a four-rules arithmetic instruction and the immediate is generated by being replaced with this rule. Inside a storage device for dividing series of instructions into plural basic blocks so as to be the operating environment of a compiler, an immediate generating memory is provided with an operand information table on the operands of instructions consisting of the respective basic blocks, flow information table on the set and reference of the register and its order and register hold value table on the hold value of the register and when registers RA and RB for storing the result of the first instruction and the result of the arithmetic operations instruction are not equal, the registers RA and RB are replaced as operands after flow analysis of these registers is performed.

Data supplied from the esp@cenet database - 12

THIS PAGE BLANK (USPTO)

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-194626

(43) 公開日 平成8年(1996)7月30日

(51) Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 9/45		7737-5B	G 0 6 F 9/44	3 2 2 E
		7737-5B		3 2 2 H

審査請求 未請求 請求項の数4 F D (全13頁)

(21) 出願番号 特願平7-22311

(22) 出願日 平成7年(1995)1月17日

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(71) 出願人 000233055

日立ソフトウェアエンジニアリング株式会社

神奈川県横浜市中区尾上町6丁目81番地

(72) 発明者 磯部 竜雄

神奈川県横浜市中区尾上町6丁目81番地

日立ソフトウェアエンジニアリング株式会社

社内

(74) 代理人 弁理士 徳若 光政

最終頁に続く

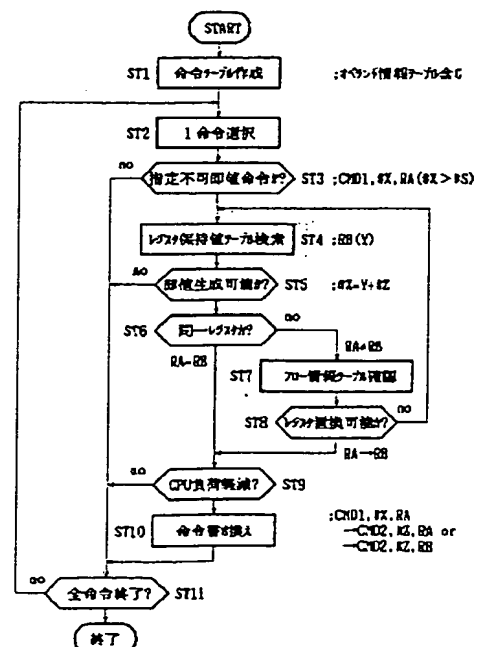
(54) 【発明の名称】 即値生成方法

(57) 【要約】 (修正有)

【目的】 即値生成のための命令追加をすることなく、制限を超えた即値の指定を可能にする。

【構成】 指定枠を超える即値#Xをオペランドとする第1の命令を超えない即値#Zを第1オペランドとし、いずれかのレジスタに残存するレジスタ保持値Yを第2オペランドとする四則演算命令に置き換え生成する。一連の命令を、複数の基本ブロックに分割し、コンパイラの動作環境となる記憶装置内に、各基本ブロックを構成する命令のオペランドに関するオペランド情報テーブルと、レジスタの設定及び参照ならびにその順序に関するフロー情報テーブルと、レジスタの保持値に関するレジスタ保持値テーブルとを含む即値生成メモリを設け、第1の命令の結果と四則演算命令の結果が格納されるレジスタRA及びRBとが同一のものでないときには、これらをフロー解析した上で、オペランドとしてのレジスタRA及びRBを置き換える。

図4 コンパイラの即値生成処理フロー



【特許請求の範囲】

【請求項1】 原始プログラムを、所定数のレジスタを具備しかつその命令の少なくともオペランド部が固定長とされるプロセッサで実行可能なオブジェクトプログラムに変換する翻訳システムにおいて、上記オペランド部の指定枠を超える第1の即値として与えられたオペランドを、上記オペランド部の指定枠を超えない第2の即値と上記レジスタのいずれかに残存するレジスタ保持値とをもとに等価的に生成することを特徴とする即値生成方法。

【請求項2】 上記第1の即値を含む第1の命令は、第1の即値を第1オペランドとし第1のレジスタを第2オペランドとするものであり、上記レジスタ保持値は、第2のレジスタに残存するものであって、上記第1の即値として与えられたオペランドの生成は、上記第1の命令を、上記第2の即値を第1オペランドとし第2のレジスタを第2オペランドとする四則演算命令に置き換えることにより実現されるものであることを特徴とする請求項1の即値生成方法。

【請求項3】 上記第1の命令の結果は、上記第1のレジスタに格納され、上記四則演算命令の結果は、上記第2のレジスタに格納されるものであって、上記翻訳システムは、第1及び第2のレジスタが同一のものでないとき、これらのレジスタの以後の振る舞いをフロー解析し、必要に応じて以後の命令の第1又は第2のオペランドとして指定されている第1のレジスタを第2のレジスタに置き換えるものであることを特徴とする請求項2の即値生成方法。

【請求項4】 上記翻訳システムにおいて、一連の命令は、その中間に分岐の始点又は終点を含まない範囲を一つの単位として複数の基本ブロックに分割されるものであって、上記翻訳システムは、その動作環境の一部となる記憶装置内に、上記基本ブロックのそれぞれを構成する一つ又は複数の命令のオペランドに関するオペランド情報テーブルと、上記所定数のレジスタの設定及び参照ならびにその順序に関するフロー情報テーブルと、上記所定数のレジスタの保持値に関するレジスタ保持値テーブルとを含む即値生成メモリを備えるものであることを特徴とする請求項1、請求項2又は請求項3の即値生成方法。

【発明の詳細な説明】**【0001】**

【産業上の利用分野】 この発明は即値生成方法に関し、例えば、原始プログラムをオペランドとして与える即値の大きさに制限のあるプロセッサで実行可能なオブジェクトプログラムに変換するためのコンパイラ等の翻訳システムならびにその実行効率の向上に利用して特に有効な技術に関するものである。

【0002】

【従来の技術】 高級言語で書かれた原始プログラムを、

マイクロコンピュータ等のプロセッサで実行可能な機械語のオブジェクトプログラムに変換するコンパイラ等の翻訳システムがある。また、命令回数や命令の種類を削減することにより、高頻度の処理を集中的に効率良く実行しうるRISC (Reduced Instruction Set Computer) 等のコンピュータシステムがある。これらのコンピュータシステムでは、その命令の特にオペランド部が固定長とされ、オペランドとして与える即値の大きさに制限が設けられる場合がある。

【0003】

【発明が解決しようとする課題】 原始プログラムをオペランドとして与える即値の大きさに制限のあるプロセッサで実行可能なオブジェクトプログラムに変換する従来の翻訳システムでは、指定枠を超えた即値をオペランドとして指定したい場合、この即値を定数定義してラベルを付与した後、所望のレジスタにロードするか、指定枠内にある複数の即値を四則演算し又は連結することで指定枠外の即値を算出した後、所望のレジスタに格納するいわゆる命令生成法によらざるを得ない。しかし、前者の方法では、定数定義やラベル付与のためにメモリの有効領域が余分に費やされるとともに、ラベル参照に際して実質的なサイクルタイムの長いメモリアクセスが必要となる。また、後者の方法では、追加された即値演算命令の格納にメモリの有効領域が余分に費やされるとともに、オブジェクトプログラムに従って動作する実行用プロセッサのステップ数が増大する。この結果、いずれにおいても実行用プロセッサの処理負担が増加し、その実行効率が低下するものである。

【0004】 この発明の目的は、定数定義によるラベル付与又はラベル参照のためのメモリアクセスあるいは即値生成命令の追加を必要とすることなく、指定枠を超える大きさの即値の指定を可能にしうる即値生成方法を提供することにある。この発明の他の目的は、その命令が固定長とされる実行プロセッサの処理負担を軽減してその実行効率を高め、メモリの使用効率を高めることにある。

【0005】 この発明の前記ならびにその他の目的と新規な特徴は、この明細書の記述及び添付図面から明らかになるであろう。

【0006】

【課題を解決するための手段】 本願において開示される発明のうち代表的なものの概要を簡単に説明すれば、次の通りである。すなわち、原始プログラムを、所定数のレジスタを具備しかつその命令の少なくともオペランド部が固定長とされるプロセッサで実行可能なオブジェクトプログラムに変換する翻訳システムにおいて、指定枠を超える第1の即値として与えられたオペランドを、指定枠を超えない第2の即値といずれかのレジスタに残存するレジスタ保持値とをもとに、しかも第1の即値をオ

ペラントとする第1の命令を、第2の即値を第1のオペラントとしレジスタ保持値を第2のオペラントとする四則演算命令に置き換えることにより生成する。このため、一連の命令を、その中間に分岐の始点又は終点を含まない範囲を一つの単位として複数の基本ブロックに分割し、翻訳システムの動作環境となる記憶装置内に、各基本ブロックを構成する一つ又は複数の命令のオペラントに関するオペラント情報テーブルと、レジスタの設定及び参照ならびにその順序に関するフロー情報テーブルと、レジスタの保持値に関するレジスタ保持値テーブルとを含む即値生成メモリを設けるとともに、第1の命令の結果が格納される第1のレジスタと四則演算命令の結果が格納される第2のレジスタとが同一のものでないときは、これらのレジスタの以後の振る舞いを即値生成メモリによりフロー解析し問題がないことを確認した上で、第1及び第2のレジスタを置き換える。

【0007】

【作用】上記した手段によれば、プログラムの実質的な機能を損いまた実行プロセッサの不正実行を招くことなく、しかも定数定義によるラベル付与又はラベル参照のためのメモリアクセスあるいは即値演算命令の追加を必要とすることなく、指定枠を超える大きさの即値の指定を可能としうる即値生成方法を実現することができる。この結果、その命令が固定長とされる実行プロセッサの処理負担を軽減してその実行効率を高め、メモリの使用効率を高めることができる。

【0008】

【実施例】図1には、この発明が適用されたコンパイラ（翻訳システム）の動作環境を説明するための一実施例のシステム構成図が示されている。また、図2及び図3には、図1のコンパイラで用いられる即値生成メモリの一実施例の記憶領域構成図が示されている。これらの図をもとに、まずこの実施例のコンパイラの動作環境と即値生成メモリの記憶領域構成ならびにその特徴について説明する。

【0009】図1において、この実施例のコンパイラは、中央処理装置（CPU）11を基本構成要素とする翻訳用コンピュータ1をその稼動環境とする。中央処理装置11には、所定の記憶容量を有する記憶装置12と、オペレータ及びコンピュータ間のマンマシンインタフェースとなるキーボード13とが結合される。また、記憶装置12には、コンパイラつまり翻訳プログラム4を格納するための領域と、この翻訳プログラム4による翻訳処理の過程で用いられる即値生成メモリ5の領域とが設けられる。翻訳用コンピュータ1は、所定の高級言語で書かれた原始プログラム3を、翻訳プログラム4に従って、実行用マイコン（マイクロコンピュータ）2で実行可能な機械語のオブジェクトプログラム6に変換する。

【0010】一方、オブジェクトプログラム6の実行環

境となる実行用マイコン2は、中央処理装置21を基本構成要素とし、この中央処理装置21には、所定の記憶容量を有するROM（リードオンリーメモリ）22が結合される。ROM22には、コンパイラにより作成されたオブジェクトプログラム6が実行プログラム7として格納される。実行用マイコン2は、ROM22内の実行プログラム7により制御され、図示されない入出力装置の制御や所定の演算処理を行う。

【0011】この実施例において、実行用マイコン2を構成するプロセッサつまり中央処理装置21は、 $p+1$ 個の汎用のレジスタ $R0 \sim Rp$ とユーザによって定義可能な $q+1$ 個のスペシャルレジスタ $SP0 \sim SPq$ とを備える。また、その命令は固定長とされ、特にその第1及び第2オペラント部はそれぞれ例えば8ビットとされる。したがって、命令の第1及び第2オペラントとして指定しうる即値の大きさは、 -128 から $+127$ までの範囲に制限される。これに対処するため、この実施例のコンパイラは、指定枠を超えた即値として指定された第1オペラントを、指定枠を超えない他の即値といずれかのレジスタに残存するレジスタ保持値とをもとに等価的に生成する即値生成機能を有し、その動作環境となる記憶装置12には、即値生成処理に供される即値生成メモリ5が設けられる。なお、コンパイラの具体的な即値生成方法については、後で詳細に説明する。

【0012】記憶装置12内に設けられた即値生成メモリ5は、図2及び図3に示されるように、命令ポインタ51、レジスタ探索ポインタ52、命令テーブル53、オペラント情報テーブル54、フロー情報テーブル55、レジスタ保持値テーブル56、レジスタフラグテーブル57、スペシャルレジスタ保持値テーブル58及びスペシャルレジスタフラグテーブル59を含む。この実施例において、プログラムを構成する一連の命令は、その中間に分岐の始点又は終点を含まない範囲を一つの単位として所定数の基本ブロック50つまり500及び501等に分割される。命令ポインタ51、レジスタ探索ポインタ52、レジスタ保持値テーブル56、レジスタフラグテーブル57、スペシャルレジスタ保持値テーブル58及びスペシャルレジスタフラグテーブル59は、すべての基本ブロック50に共通に設けられ、命令テーブル53、オペラント情報テーブル54及びフロー情報テーブル55は、各基本ブロック50に対応して個別に設けられる。

【0013】ここで、即値生成メモリ5の命令テーブル53は、各基本ブロックの命令テーブルを順次リンクさせるためのリンク情報 $LINK0$ 及び $LINK1$ 等と、その中間に分岐の始点又は終点を含まない任意数の命令 $INST00 \sim INST03$ 等ならびに $INST10 \sim INST13$ 等とを保持する。また、オペラント情報テーブル54は、命令テーブル53の対応する命令 $INST00 \sim INST03$ 等ならびに $INST10 \sim INST$

T13等の第1オペランド541及び第2オペランド542を保持し、その内容は、レジスタR0～RpあるいはスペシャルレジスタSR0～SRqであったり、即値IMMDであったりする。

【0014】一方、フロー情報テーブル55は、設定レジスタ情報REGS LIST0及びREGS LIST1等、参照レジスタ情報REGR LIST0及びREGR LIST1等、設定参照順序情報S/R ORD0及びS/R ORD1等ならびに分岐先情報BRANCH0及びBRANCH1等を含む。このうち、設定レジスタ情報REGS LIST0及びREGS LIST1等は、その各ビットがレジスタR0～RpあるいはスペシャルレジスタSR0～SRqに対応され、対応する基本ブロックにおいて対応するレジスタ又はスペシャルレジスタに何らかの値が設定されているとき選択的にセットされる。同様に、参照レジスタ情報REGR LIST0及びLISTR LIST1等は、その各ビットがレジスタR0～RpあるいはスペシャルレジスタSR0～SRqに対応され、対応する基本ブロックにおいて対応するレジスタ又はスペシャルレジスタがいずれかの命令によって参照されるととき選択的にセットされる。

【0015】さらに、レジスタ保持値テーブル56は、対応するレジスタR0～Rpに設定され残存する値を保持する。また、レジスタフラグテーブル57は、その各ビットがレジスタR0～Rpに対応され、メモリ内容が転送又は演算の対象となりあるいはアドレスが設定されたために対応するレジスタR0～Rpの保持値が翻訳時に未知の値となったとき、選択的にセットされる。同様に、スペシャルレジスタ保持値テーブル58は、対応するスペシャルレジスタSR0～SRqに設定され残存する値を保持する。また、スペシャルレジスタフラグテーブル58は、その各ビットがスペシャルレジスタSR0～SRqに対応され、メモリ内容が転送又は演算の対象となりあるいはアドレスが設定されたために対応するスペシャルレジスタSR0～SRqの保持値が翻訳時に未知の値となったとき、選択的にセットされる。

【0016】命令ポインタ51は、後述するように、コンパイラによる翻訳処理の過程で、命令テーブル53に格納された命令INST00～INST03等ならびにINST10～INST13等を順次指定し、指定枠を超えた即値を含む命令に対して即値生成処理を選択的に施すためのポインタとなる。また、レジスタ探索ポインタ52は、この即値生成処理の過程で、レジスタ保持値テーブル56ならびにスペシャルレジスタ保持値テーブル58を順次指定し、指定枠を超えた即値を等価的に生成すべくレジスタ保持値を探索するためのポインタとなる。

【0017】図4には、図1のコンパイラによる即値生成処理の一実施例のフロー図が示されている。また、図

5には、図1のコンパイラの処理対象となるプログラムのうち即値生成が必要な部分の一実施例の部分的な命令リストが示され、図6には、そのもう一つの実施例の部分的な命令リストが示されている。これらの図をもとに、この実施例のコンパイラによる即値生成処理の具体的方法ならびにその特徴について説明する。なお、図5には、指定枠を超えた即値を含む命令の第2オペランドとなるレジスタ（第1のレジスタ）と即値生成に供されるレジスタ保持値を保持するレジスタ（第2のレジスタ）とが同一レジスタR1である場合が例示され、図6には、これらのレジスタが同一レジスタではなく異なるレジスタR1及びR2である場合が例示される。また、図5及び図6には、(a)として原始プログラムに近い翻訳処理の中間段階つまりは即値生成処理が施されない状態におけるオブジェクトプログラムが、(b)として(a)のプログラムを従来のコンパイラで処理した後のオブジェクトプログラムが、また(c)として(a)のプログラムをこの発明が適用されたコンパイラで即値生成処理した後のオブジェクトプログラムがそれぞれ示されている。さらに、各命令リストの左外側には、各命令のプログラム行番号が付記され、その右外側には、各命令に関するコメントが付記されているので、説明にあわせて参照されたい。

【0018】図4において、この実施例のコンパイラによる即値生成処理は、ステップST1つまりオペランド情報テーブル54を含む命令テーブル53の作成によって開始される。このとき、翻訳過程にあるプログラムを構成する一連の命令は、前述のように、その中間に分岐の始点又は終点を含まない範囲を一つの単位として複数の基本ブロック50に分割され、翻訳用コンピュータ1の記憶装置12の即値生成メモリ5には、これらの基本ブロック50に対応して作成された命令テーブル53及びオペランド情報テーブル54が用意される。

【0019】命令テーブル53及びオペランド情報テーブル54の作成が終了すると、コンパイラは、ステップST2により、最初的基本ブロック500から命令ポインタ51によって指定される一つの命令を選択し、ステップST3により、選択された命令が指定枠を超えた即値を第1オペランドとして含むいわゆる指定不可即値命令であるかどうかを判定する。このとき、命令は、コメントとして付記されるように、コマンドCMD1と、第1オペランドとなる即値#X（ここで、#はオペランドがXなる十進数の即値からなることを表す。以下同様）と、第2オペランドとなるレジスタRAとを含み、この命令が指定不可即値命令である場合、第1オペランドである即値#Xはその指定枠#Sより大きな値となる。この実施例において、実行用マイコン2が実行しうる命令の第1オペランドは、前述のように、8ビットとされ、その最大値#Sは+127とされる。なお、各命令で第2オペランドとして指定されるレジスタRA等は、当該

命令の実行結果を格納するためのいわゆるデスティネーションオペランドともなる。

【0020】ステップST3において、選択された命令が指定枠を超える即値を含まず指定不可即値命令ではない場合、コンパイラは、ステップST11により、すべての命令に関する選択処理が終了したかを判定し、まだ未選択の命令が存在する場合には、ステップST2に戻り、未選択の命令が存在しない場合には、一連の即値生成処理を終結する。なお、コンパイラは、ステップST11の判定処理に移行する直前で、図示されないフロー情報テーブル55、レジスタ保持値テーブル56、レジスタフラグテーブル57、スペシャルレジスタ保持値テーブル58及びスペシャルレジスタフラグテーブル59の更新処理を行う。すなわち、コンパイラは、選択された命令によりレジスタR0～RpあるいはスペシャルレジスタSR0～SRqに何らかの値が設定され又は参照されるのをモニタし、フロー情報テーブル55の設定レジスタ情報及び参照レジスタの対応するビットをセットするとともに、各レジスタに設定された値を可能な限りレジスタ保持値テーブル56又はスペシャルレジスタ保持値テーブル58に登録する。また、選択された命令によりメモリ内容が転送又は演算の対象となりあるいはアドレスが設定されたために対応するレジスタR0～RpあるいはスペシャルレジスタSR0～SRqの保持値が不定となったときには、レジスタフラグテーブル57又はスペシャルレジスタフラグテーブル59の対応するビットをセットする。

【0021】一方、ステップST3において、選択された命令が指定枠を超える即値#X（第1の即値）を含む指定不可即値命令であった場合、コンパイラは、ステップST4により、レジスタ保持値テーブル56を参照し、その保持値Yと指定枠を超えない即値#Z（第2の即値）とにより指定枠を超えた即値#Xを等価的に生成できそうなレジスタを探索する。すなわち、例えば、図5（a）のプログラム行L23の移動命令『MOV #128, R1』のように、選択された命令の第1オペランドが指定枠を超える即値#Xつまり“128”であった場合、コンパイラは、その前に実行された移動命令『MOV #127, R1』によりレジスタR1に設定された指定枠内の値Yつまり“127”に着目する。そして、

$$128 - 127 = 1$$

であることから、指定枠内の即値#Zとして、

$$\#Z = 1$$

を算出する。この結果、ステップST5において、

$$\#X = Y + \#Z$$

つまり、例えば『ADD #Z, R1』なる加算命令により“128”なる指定枠外の即値#Xが等価的に生成可能であることを判定する。

【0022】即値生成可能であることを判定したコンパ

イラは、ステップST6により、もとの命令『CMD1

#X, RA』の第2オペランドつまりデスティネーションオペランドとなるレジスタRA（第1のレジスタ）と、レジスタ保持値Yを保持するレジスタRB（第2のレジスタ）とが同一レジスタであるかどうかを判定する。その結果、レジスタRA及びRBが同一レジスタである場合には、ステップST9において、命令置き換えにより実行用マイコン2の中央処理装置21の処理負荷が軽減されるかどうかを判定し、負荷軽減の見込みがある場合には、ステップST10により、もとの命令『CMD1 #X, RA』（第1の命令）を新しい命令『CMD2 #Z, RA』（第2の命令）に置き換える。これにより、例えば図5（a）のプログラム行L23の移動命令『MOV #128, R1』は、図5（c）のプログラム行L43の加算命令『ADD #1, R1』にそのまま1対1で置き換えられ、同一の実行結果が得られるものとなる。

【0023】なお、ステップST6において、レジスタRA及びRBが同一レジスタでないと判定された場合、コンパイラは、ステップST7により、フロー情報テーブル55の設定レジスタ情報及び参照レジスタ情報ならびに設定参照順序情報を参照し、ステップST8により、レジスタRA及びRBの置き換えが可能であるかどうかを判定する。すなわち、コンパイラは、フロー情報テーブル55の設定レジスタ情報及び参照レジスタ情報ならびに設定参照順序情報をもとに、レジスタRA及びRBの以後の振る舞いをフロー解析する。そして、レジスタRAが、

（1）対応する命令以後の同一基本ブロック内あるいはフロー情報テーブル55の分岐先情報BRANCH0及びBRANCH1等をもとにたぐりうるすべての基本ブロック内で設定されていること

（2）次に設定されるまでに参照がないこと

（3）参照があったとしてもその基本ブロックに分岐するのは当該基本ブロックのみであること

なる条件を満たし、しかもレジスタBが、

（1）対応する命令以後の同一基本ブロック内あるいはフロー情報テーブル55の分岐先情報BRANCH0及びBRANCH1等をもとにたぐりうるすべて基本ブロック内で設定されていること

（2）次に設定されるまでに参照がないこと

なる条件を満たすことを確認した上で、ステップST10による命令の書き換えを実行するとともに、新しい命令『CMD2 #Z, RA』の第2オペランドをレジスタRBに置き換えて『CMD2 #Z, RB』とし、以後の命令でオペランドとして指定されるレジスタRAもレジスタRBに置き換える。なお、このフローの中に図示してはいないが、命令1個1個処理してゆくと同時にレジスタ保持値テーブルを逐次更新してゆく処理が必要なことは言うまでもない。

【0024】これらのことから、例えば、図6(a)の指定枠を超える即値#250を含むプログラム行L53の移動命令『MOV #250, R2』に着目した場合、プログラム行L51の移動命令『MOV #127, R1』の実行結果としてレジスタR1に設定された値“127”を利用して即値の生成は可能であるが、この値“127”を保持するレジスタR1がもとの移動命令『MOV #250, R2』の第2オペランドとなるレジスタR2とは異なるため、これらのレジスタの以後の振る舞いをフロー解析した上で、もとの移動命令は図6(c)のプログラム行L73の加算命令『ADD #123, R1』に置き換えられる。言うまでもなく、レジスタR1の保持値は“127”であり、この加算命令の結果“250”は同じレジスタR1に格納される。また、図6(a)のプログラム行L54の減算命令『SUB R2, R3』の第1オペランドとして指定されるレジスタR2は、上記レジスタの置き換えを受けてレジスタR1に置き換えられる。この結果、プログラムの本来の機能を維持できるとともに、レジスタ置き換えにともなう実行用マイコン2の不正実行を防止することができる。

【0025】ところで、即値生成機能を持たない従来のコンパイラでは、図5(a)の移動命令『MOV #128, R1』あるいは図6(a)の移動命令『MOV #250, R2』のような指定枠を超える即値を直接指定できない。したがって、このような即値#128又は#250を指定したい場合、図5(b)のプログラム行L35の定義命令『DCL 128』あるいは図6

(b)の定義命令『DCL 250』によってこれを定義し、『LABEL1』又は『LABEL2』なるラベルを付与した後、もとの移動命令を図5(b)のプログラム行33の移動命令『MOV LABEL1, R1』あるいは図6(b)のプログラム行63の移動命令『MOV LABEL1, R1』に書き換えるか、もとの移動命令の直前に128又は250を結果とする四則演算命令を追加し、もとの移動命令の第1オペランドを四則演算命令のデスティネーションオペランドとなるレジスタに書き換える必要がある。しかし、定数定義を行う図5(b)及び図6(b)の方法では、ラベル付与のためにメモリの有効領域が余分に費やされるとともに、プログラム行L33又はL63のラベル参照に際して実質的なサイクルタイムの長いメモリアクセスが必要となり、四則演算命令を追加する方法では、追加命令の格納にメモリの有効領域が余分に費やされるとともに、実行用マイコン2におけるオブジェクトプログラム6のステップ数が増大する。この結果、いずれの場合も実行用マイコン2の処理負担が増大し、その実行効率が低下する。

【0026】ところが、この発明が適用されたコンパイラの場合、指定枠を超えた即値を含む移動命令『MOV #128, R1』及び『MOV #250, R2』

は、前述のように、指定枠を超えない即値#1又は#123とレジスタR1の保持値127とによる加算命令『ADD #1, R1』又は『ADD #123, R2』にそのまま1対1で置き換えられるとともに、これらの命令に関与するレジスタの以後の振る舞いがフロー解析され、必要に応じてオペランドの置き換えが行われる。これらのことから、プログラムの実質的な機能を損いまた実行用マイコン2の不正実行を招くことなく、しかも定数定義によるラベル付与又はラベル参照のためのメモリアクセスあるいは即値生成のための命令追加を必要とすることなく、指定枠を超える大きさの即値の指定を可能としうる即値生成方法を実現できる。この結果、その命令が固定長とされる実行用マイコン2の処理負担を軽減してその実行効率を高め、メモリの使用効率を高めることができる。

【0027】以上の実施例から得られる作用効果は、下記の通りである。すなわち、

(1) 原始プログラムを、所定数のレジスタを具備しかつその命令の少なくともオペランド部が固定長とされるプロセッサで実行可能なオブジェクトプログラムに変換する翻訳システムにおいて、指定枠を超える第1の即値として与えられたオペランドを、指定枠を超えない第2の即値といずれかのレジスタに残存するレジスタ保持値とをともに、しかも第1の即値をオペランドとする第1の命令を、第2の即値を第1のオペランドとしレジスタ保持値を第2のオペランドとする四則演算命令に置き換えることにより生成することで、定数定義によるラベル付与又はラベル参照のためのメモリアクセスあるいは即値生成のための命令追加を必要とすることなく、指定枠を超える大きさの即値の指定を可能としうる即値生成方法を実現することができるという効果が得られる。

【0028】(2) 上記(1)項において、一連の命令を、その中間に分岐の始点又は終点を含まない範囲を一つの単位として複数の基本ブロックに分割し、翻訳システムの動作環境となる記憶装置内に、各基本ブロックを構成する命令のオペランドに関するオペランド情報テーブルと、レジスタの設定及び参照ならびにその順序に関するフロー情報テーブルと、レジスタの保持値に関するレジスタ保持値テーブルとを含む即値生成メモリを設けるとともに、第1の命令の結果が格納される第1のレジスタと四則演算命令の結果が格納される第2のレジスタとが同一のものでないときには、これらのレジスタの以後の振る舞いを即値生成メモリによりフロー解析し問題がないことを確認した上で、選択的に第1及び第2のレジスタを置き換えることで、プログラムの本来の機能を保ち、プロセッサ等の不正実行を防止することができるという効果が得られる。

(3) 上記(1)項及び(2)項により、その命令が固定長とされるプロセッサ等の処理負担を軽減してその実行効率を高めることができるとともに、そのメモリの使

用効率を高めることができるという効果が得られる。

【0029】以上、本発明者によってなされた発明を実施例に基づき具体的に説明したが、この発明は、上記実施例に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能であることは言うまでもない。例えば、図1において、翻訳用コンピュータ1及び実行用マイコン2のブロック構成は、種々の実施形態を採りうる。図2及び図3において、即値生成メモリ5の構成は、この実施例による制約を受けないし、その利用方法も任意である。図4において、コンパイラによる即値生成処理は、基本的なアルゴリズムが変わらないことを条件に、種々の処理フローを採りうる。図5及び図6において、命令の構成、オペランド等の組み合わせ及び順序等は、種々の実施形態が考えられよう。即値生成のための演算命令には、加算命令以外の四則演算命令を利用してもよいし、四則演算命令以外の命令を利用することもできる。また、実行用マイコン2は、必ずしもスペシャルレジスタSR0～SRqを備えることを必須条件とはしない。さらに、即値生成の対象となるオペランドは、第1オペランドに限定されないし、即値の具体的な指定枠や命令の種別及び名称等も、この実施例の限りではない。

【0030】以上の説明では、主として本発明者によってなされた発明をその背景となった利用分野であるコンパイラに適用した場合について説明したが、それに限定されるものではなく、例えばアセンブラやオブティマイザ等の各種翻訳システムにも適用できる。この発明は、少なくとも原始プログラムをオペランドとして与える即値の大きさに制限のあるプロセッサで実行可能なオブジェクトプログラムに変換するための翻訳システムならびにその即値生成方法に広く適用できる。

【0031】

【発明の効果】本願において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば、下記の通りである。すなわち、原始プログラムを、所定数のレジスタを具備しかつその命令の少なくともオペランド部が固定長とされるプロセッサで実行可能なオブジェクトプログラムに変換する翻訳システムにおいて、指定枠を超える第1の即値として与えられたオペランドを、指定枠を超えない第2の即値といずれかのレジスタに残存するレジスタ保持値とをもとに、しかも第1の即値をオペランドとする第1の命令を、第2の即値を第1のオペランドとしレジスタ保持値を第2のオペランドとする四則演算命令に置き換えることにより生成する。このため、一連の命令を、その中間に分岐の始点及び終点を含まない範囲を一つの単位として複数の基本ブロックに分割し、翻訳システムの動作環境となる記憶装置内に、各基本ブロックを構成する命令のオペランドに関するオペランド情報テーブルと、レジスタの設定及び参照ならびにその順序に関するフロー情報テーブルと、レジスタの

保持値に関するレジスタ保持値テーブルとを含む即値生成メモリを設けるとともに、第1の命令の結果が格納される第1のレジスタと四則演算命令の結果が格納される第2のレジスタが同一のものでないときには、これらのレジスタの以後の振る舞いを即値生成メモリによりフロー解析し問題がないことを確認した上で、第1及び第2のレジスタを置き換える。これにより、プログラムの本来の機能を損いまたプロセッサの不正実行を招くことなく、しかも定数定義によるラベル付与又はラベル参照のためのメモリアクセスあるいは即値生成のための命令追加を必要とすることなく、指定枠を超える大きさの即値の指定を可能としうる即値生成方法を実現することができる。この結果、その命令が固定長とされるプロセッサ等の処理負担を軽減してその実行効率を高め、メモリの使用効率を高めることができる。

【図面の簡単な説明】

【図1】この発明が適用された即値生成方法を採るコンパイラの動作環境を説明するための一実施例を示すシステム構成図である。

【図2】図1のコンパイラで用いられる即値生成メモリの一実施例を示す部分的な記憶領域構成図である。

【図3】図1のコンパイラで用いられる即値生成メモリの一実施例を示す他の部分的な記憶領域構成図である。

【図4】図1のコンパイラによる即値生成処理の一実施例を示すフロー図である。

【図5】図1のコンパイラの処理対象となるプログラムのうち即値生成を必要とする部分の一実施例を示す命令リストである。

【図6】図1のコンパイラの処理対象となるプログラムのうち即値生成を必要とする部分の他の一実施例を示す命令リストである。

【符号の説明】

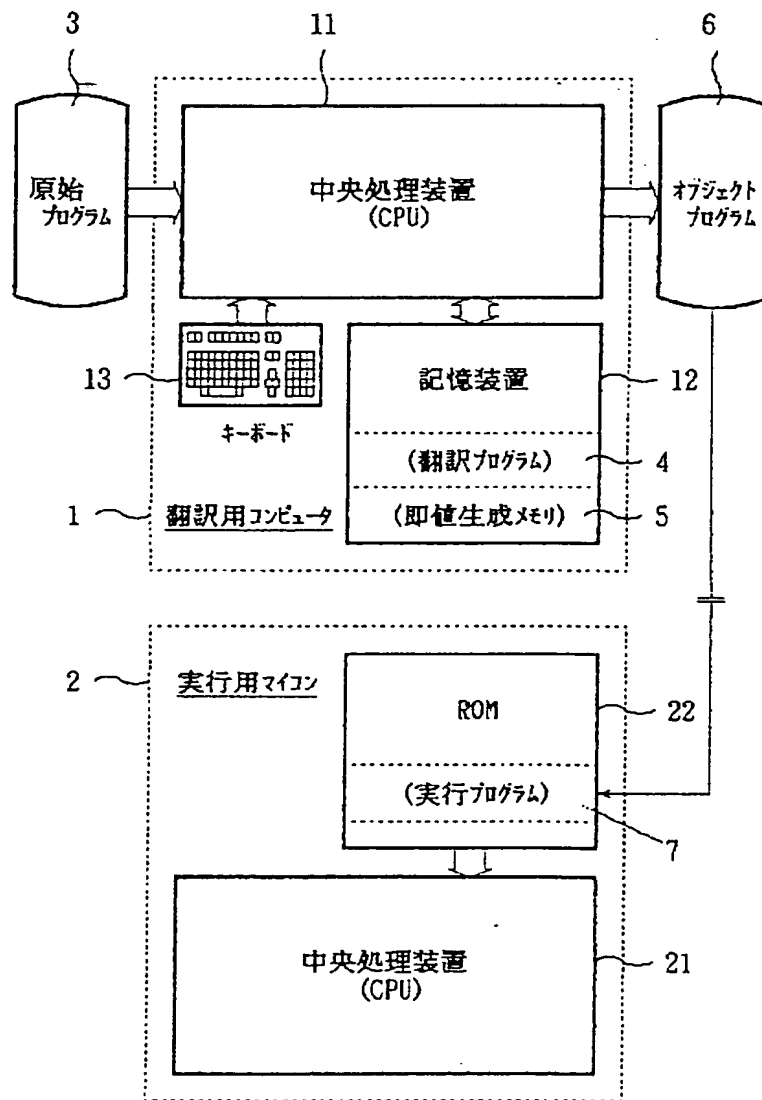
1・・・翻訳用コンピュータ、2・・・実行用マイコン（マイクロコンピュータ）、11、21・・・中央処理装置、12・・・記憶装置、22・・・ROM（リードオンリーメモリ）、13・・・キーボード、3・・・原始プログラム、4・・・翻訳プログラム（コンパイラ）、5・・・即値生成メモリ、6・・・オブジェクトプログラム、7・・・実行プログラム、50、500～501・・・基本ブロック、51・・・命令ポインタ、52・・・レジスタ探索ポインタ53・・・命令テーブル、LINK0～LINK1・・・リンク情報、INST00～INST03ないしINST10～INST13・・・命令、54・・・オペランド情報テーブル、541・・・第1オペランド、542・・・第2オペランド、IMMD・・・即値、R0～Rp・・・レジスタ、SR0～SRq・・・スペシャルレジスタ、55・・・フロー情報テーブル、REGS LIST0、REGS LIST1・・・設定レジスタ情報、REGR LIST0、REGR LIST1・・・参照レジスタ情

報、S/RORD0, S/R ORD1・・・設定参照
順序情報、BRANCH0, BRANCH1・・・分岐
先情報。56・・・レジスタ保持値テーブル、REG0
～REGp・・・レジスタ保持値、57・・・レジスタ
フラグテーブル、REGF LIST・・・レジスタフ
ラグリスト、58・・・スペシャルレジスタ保持値テー
ブル、SPE REG0～SPE REGq・・・スベ

シャルレジスタ保持値、59・・・スペシャルレジスタ
フラグテーブル、SPE REGF LIST・・・ス
ペシャルレジスタフラグ情報。ST1～ST11・・・
処理ステップ、CMD1～CMD2・・・コマンド、R
A～RB・・・レジスタ、#S, #X, #Y, #Z・・・
即値。L21～L24ないしL71～L74・・・プ
ログラム行。

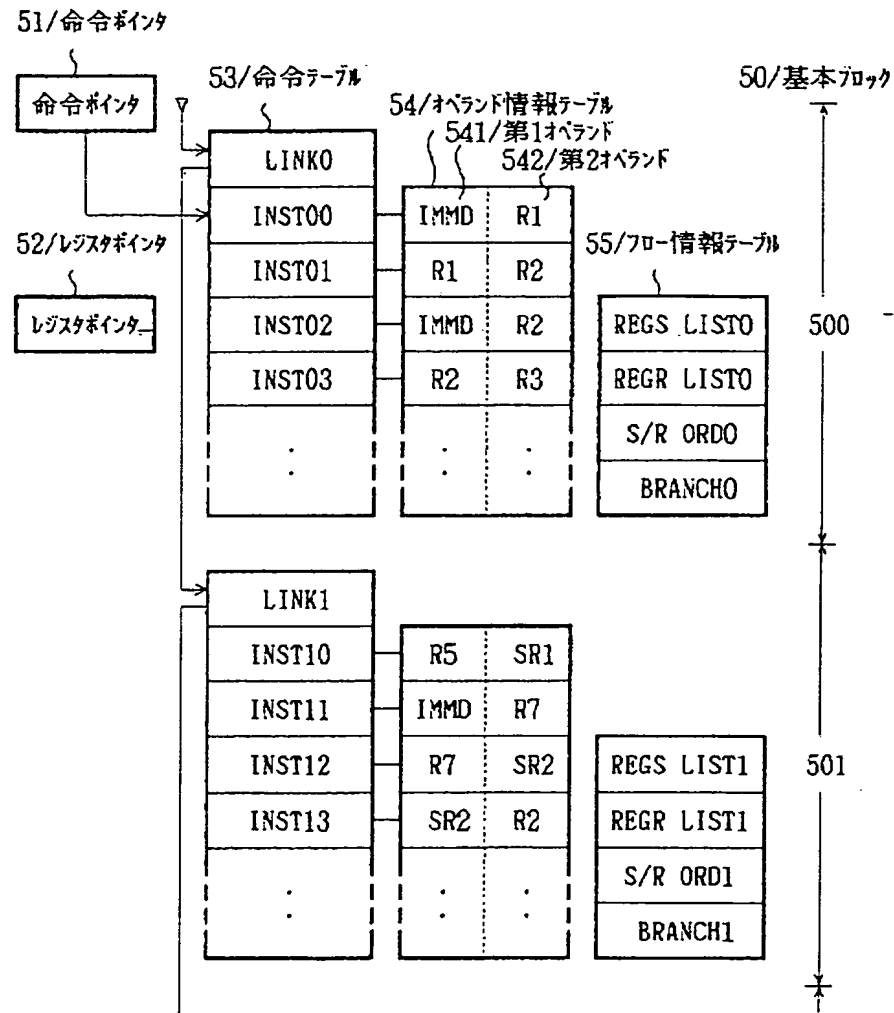
【図1】

図1 コンパイラの動作環境



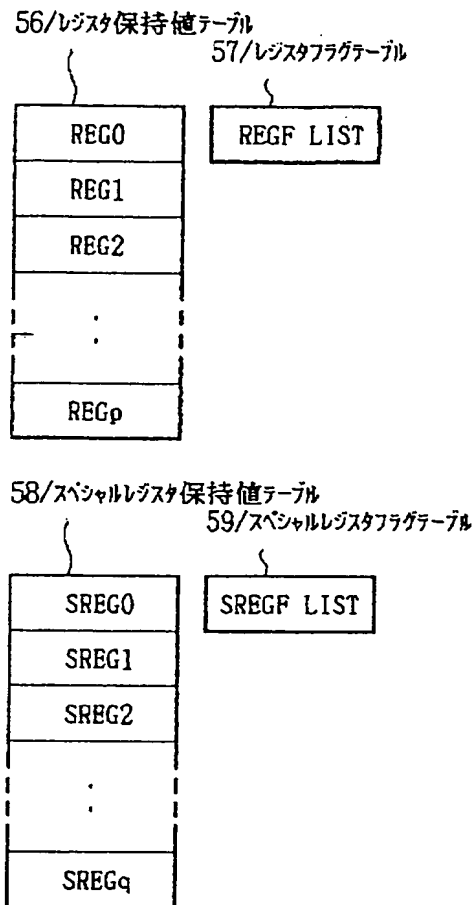
【図2】

図2 即値生成メモリの部分的な記憶領域構成 (その1)



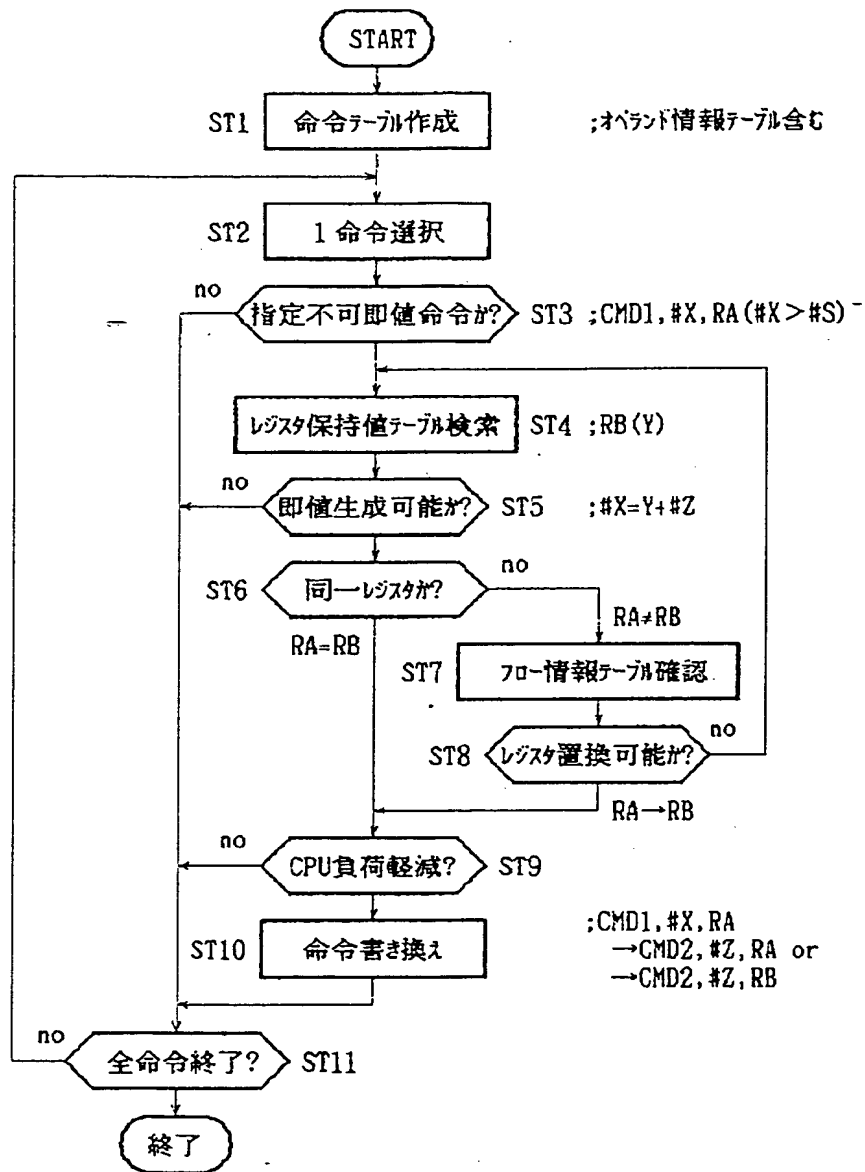
【図3】

図3 即値生成メモリの部分的な記憶領域構成 (その2)



【図4】

図4 コンパイラの即値生成処理フロー



【図5】

図5 即値生成が必要なプログラムの一例 (レジスタ一致)

(a) 翻訳の中間段階におけるオブジェクトプログラム

⋮	⋮		
L21	MOV	#127 ,R1	;127→R1
L22	ADD	R1 ,R3	;R1+R3→R3
⋮	⋮		
L23	MOV	#128 ,R1	;128→R1/指定枠オーバー
L24	SUB	R1 ,R3	;R1-R3→R3
⋮	⋮		

(b) 従来のコンパイラによるオブジェクトプログラム

⋮	⋮		
L31	MOV	#127 ,R1	;127→R1
L32	ADD	R1 ,R3	;R1+R3→R3
⋮	⋮		
L33	MOV	LABEL1,R1	;LABEL1→R1/メモリ参照発生
L34	SUB	R1 ,R3	;R1-R3→R3
⋮	⋮		
L35	LABEL1:DCL	128	;LABEL1の定義
⋮	⋮		

(c) 本発明のコンパイラによるオブジェクトプログラム

⋮	⋮		
L41	MOV	#127 ,R1	;127→R1
L42	ADD	R1 ,R3	;R1+R3→R3
⋮	⋮		
L43	ADD	#1 ,R1	;1+R1→R1/命令音字換え
L44	SUB	R1 ,R3	;R1-R3→R3
⋮	⋮		

【図6】

図6 即値生成が必要なプログラムの一例 (レジスタ不一致)

(a) 翻訳の中間段階におけるオブジェクトプログラム

⋮	⋮		
L51	MOV	#127 ,R1	:127→R1
L52	ADD	R1 ,R3	:R1+R3→R3
⋮	⋮		
L53	MOV	#250 ,R2	:250→R2/指定枠オーバー
L54	SUB	R2 ,R3	:R2-R3→R3
⋮	⋮		

(b) 従来のコンパイラによるオブジェクトプログラム

⋮	⋮		
L61	MOV	#127 ,R1	:127→R1
L62	ADD	R1 ,R3	:R1+R3→R3
⋮	⋮		
L63	MOV	LABEL2,R2	:LABEL2→R2/メモリ参照発生
L64	SUB	R2 ,R3	:R2-R3→R3
⋮	⋮		
L65	LABEL2:DCL	250	:LABEL2の定義
⋮	⋮		

(c) 本発明のコンパイラによるオブジェクトプログラム

⋮	⋮		
L71	MOV	#127 ,R1	:127→R1
L72	ADD	R1 ,R3	:R1+R3→R3
⋮	⋮		
L73	ADD	#123 ,R1	:123+R1→R1/命令書き換え
L74	SUB	R1 ,R3	:R1-R3→R3/オペランド書き換え
⋮	⋮		

フロントページの続き

(72)発明者 柏木 有吾
 東京都小平市上水本町5丁目20番1号 株
 式会社日立製作所半導体事業部内

THIS PAGE BLANK (USPTO)